A Testbed for Managing Dynamic Mixed Workloads

Stefan Krompass[TUM], Harumi Kuno[HPL], Janet L. Wiener[HPL], Kevin Wilkinson[HPL],
Umeshwar Dayal[HPL], Alfons Kemper[TUM]

[TUM]Technische Universität München,
Munich, Germany
{firstname.lastname}@in.tum.de

[HPL]Hewlett-Packard Laboratories
Palo Alto, CA, USA
{firstname.lastname}@hp.com

ABSTRACT

Workload management for operational business intelligence (BI) databases is difficult. Queries vary widely in length and objectives. Resource contention is difficult to predict and to control as dynamically-arriving, long, analyst queries compete for resources with ongoing online-transaction processing (OLTP) queries and batch report queries. Currently, administrators struggle to choose workload management policies and set their thresholds manually. The goal of our project is a software framework to make the management of such mixed workloads easier. Our framework includes a policy controller that tunes workload management policies automatically to meet workload objectives. This demonstration of our system illustrates (1) the difficulty of managing a BI database workload and (2) the benefits of tuning policies automatically and individually for each service class of queries in a workload. In addition, our demonstrator is a useful research tool for understanding how policies and a policy controller adapt as the system state changes under a mixed workload.

In our demo, the participant plays the administrator and tunes the policies for a variety of difficult-to-manage workloads as they execute. These policies include admission control, scheduling, and execution control policies. We visualize the policies, the user objectives, and the load on the system components (CPUs, memory, disks) during execution, which helps the participant see whether objectives are being met and make appropriate policy decisions. At the end of each workload, the participant is given the opportunity to compare how their policies met workload objectives versus policies determined by our automatic policy controller.

1. INTRODUCTION

Enterprise business intelligence (BI) database systems are moving towards supporting online, operational decision making at all levels in the enterprise [1, 7]. A single operational BI system might need to maintain consistent throughput for OLTP-style queries generated by cashiers ringing up sales while also providing fast response times for queries submitted by financial analysts seeking an immediate answer and reliable completion times for queries kicked off by monthly status report generation. In general, operational BI systems must support mixed workloads composed of different types of queries with different objectives and dynamic arrival rates.

VLDB '09, August 24-28, 2009, Lyon, France

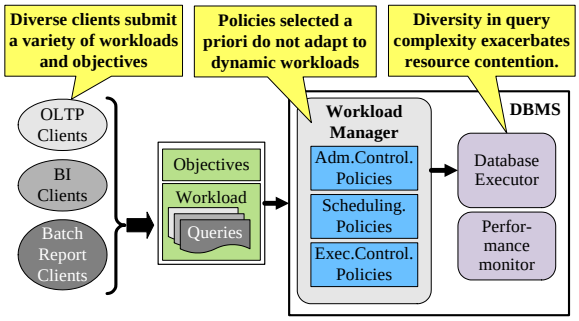

Figure 1: Architecture of management today: Traditional workload management can compensate for short-term problems, but cannot adapt policies in the presence of longer-term problems caused by (dynamic) changes in the workload.

Figure 1 shows the conventional workload management architecture for such a system. The system is pre-configured to support some number of service classes. Each service class is associated with a static set of workload management policies and objectives. As queries from various service classes arrive, the workload management system uses these statically selected policies to make admission control, scheduling, and execution control decisions. As the query mix changes, a database administrator must recognize when and which new polices are needed, and manually adjust the system configuration. For example, if a set of high-priority OLTP queries begins to arrive while an analyst is running resource-intensive ad hoc queries, then a new admission control policy that sets absolute thresholds on resource usage may be necessary for the analyst's queries.

It can be extremely difficult to determine the best policy and threshold values to meet the objectives of all queries in the system. Once policies and thresholds have been set, administrators must constantly monitor system performance and make manual adjustments. This administration greatly increases the cost of ownership. However, members of the academic database community who have not worked as database administrators typically do not understand how difficult these choices are or what makes them hard. This demonstration gives participants a hands-on experience that will illustrate the difficulty and the importance of these challenges, and, we hope, encourage database researchers to apply their particular skills to advancing the state of the art in database workload management.

In Figure 2, we propose a new architecture with three feedback loops to guide (or automate) the administrator's tasks.

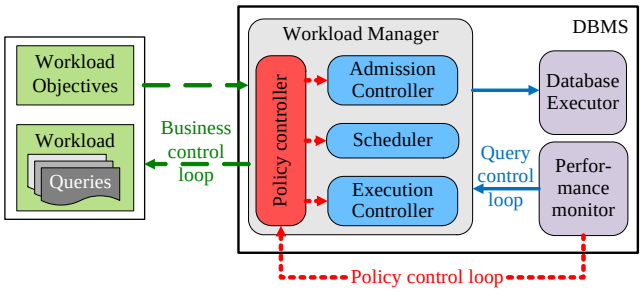

Figure 2: The three feedback control loops in workload management: the query control loop (solid blue arrows), the policy control loop (dotted red arrows) that is the focus of this demo, and the business control loop (dashed green arrows).

- The *query control loop* uses feedback from performance monitoring and applies traditional workload management policies to decide which queries to admit, when to run them, and if and when to kill them. It can compensate for incomplete knowledge about query resource usage and arrival times. The admission controller, scheduler, and execution controller implement this loop.
- The *policy control loop* responds to the dynamic changes in mixed workloads common to operational BI systems. The policy controller defines and assigns admission control, scheduling, and execution control policies, and adapts them to accommodate changes in the queries or their objectives.
- The *business control loop* identifies situations when the underlying system is not appropriate for the workload. In such cases, either the workload objectives may be redefined (through revised service level agreements) or the database system may be resized to better accommodate the workload.

Most commercial database workload management systems implement some admission control, scheduling, and execution control policies. Our earlier work [3, 2] explores the effectiveness of several static admission control, scheduling, and execution control policies in the query control loop. This loop can remedy short-term fluctuations in the workload such as a single long-running query that unexpectedly hogs resources or lock contention that creates a convoy. However, it cannot change the policies themselves.

We focus here on the policy control loop. Our demo challenges a participant to make the policy controller's decisions. We hope to demonstrate that making these decisions is, in fact, difficult as query arrival rates and objectives (e.g., minimize response time, complete within 45 minutes, or maintain query throughput) change.

In our current work, we are designing a novel *policy controller* that detects when existing workload management policies will fail to meet objectives, chooses the appropriate corrective policies and thresholds for the current workload, and notifies the workload management components of the new policies. In our demo, the participant plays against our policy controller during the execution of a challenging and changing workload. The participant tries to manually tune the workload management policies to meet service class objectives using the database administrator interface. This interface visualizes the policies, the objectives, and the load on the system components (CPUs, memory, disks). At the end of each workload, the demo contrasts the participant's ability to meet the workload objectives with that of the system's automatic policy controller. In the future, we plan to examine the business control loop, which must recognize when the system cannot satisfy the workload objectives.

2. RELATED WORK

Our focus in this demonstration is on the automated policy control loop. Although other researchers consider the problem of resource allocation with regard to service objectives or adjust policy thresholds based on runtime feedback, our work is unique in that our policy control loop automates both selecting and setting thresholds for workload management policies for dynamically changing workloads with heterogeneous objectives.

Pang *et al.* [5] propose a *Priority Memory Management* algorithm that uses feedback to adjust both scheduling and execution control policies while running workloads composed of multiple service classes. However, their workloads are homogeneous, and contain only deadline-driven queries against real-time database systems. They do not consider multiple service classes with different types of objectives (e. g., one deadline-driven and one defined in terms of throughput). Also, they adjust policy thresholds, but do not make fundamental changes to the policies themselves.

Niu *et al.* [4] have built a framework that uses feedback from a performance monitor to implement something similar to our policy control loop that manages a mixed OLTP/OLAP workload. However, their workload composition is known a priori, and their control loop sets policy thresholds but does not change the policies that apply to the various classes of queries. Furthermore, their query control loop does not include execution control actions.

Teradata Dynamic Workload Manager [6] allows administrators to specify explicitly-named states that reflect both the system condition (e.g., "normal" or "degraded") as well as the type of work the database system is expected to perform (e.g., "daily loads" or "operational queries"). Each state has associated workload management policies, and a state change may activate different workload management policies. Administrators can provide fixed time windows in which a state, and thus a set of workload management policies, is active. In addition, the administrator can set up the workload manager to detect specific changes in system conditions or workload composition, and to trigger state changes based on these events. Thus, although the Teradata Dynamic Workload Manager is capable of selecting workload management policies automatically at runtime, given a set of anticipated conditions, the thresholds used by policies are also preset and are not adjusted automatically to respond to runtime conditions.

3. DEMONSTRATION SCENARIO

We first demonstrate that the performance of a mixed workload can be difficult to control, even when the individual queries in the workload are well understood. Second, we demonstrate that our policy control loop can restore system balance, e. g., by applying resource usage policies when changes to the executing workload cause an overload.

In our demonstration, a database administrator attempts to manage a running workload so as to meet its service level objectives. We periodically introduce complications into the running workload. The workload is executed by our simulator of a multi-node shared-nothing database system, using the detailed plans produced by a commercial query optimizer as input, supplemented by query execution statistics, if available. The simulator lets us demonstrate in minutes the management of workloads that run for hours.

We provide four different scenarios in our demo. We present here one example scenario. The queries in our scenarios are divided into three service classes:

- The "CEO" service class has highest-priority, hand-written, ad hoc queries written on behalf of a company executive. They arrive at unpredictable times. The only information the

Figure 3: Policy control window.

workload manager has about the expected resource usage and behavior of these queries is the optimizer's cost estimates. The objective for each query is to complete it promptly — as long as its cost estimates are accurate. In terms of workload management policies, (1) each query should be immediately admitted and scheduled, (2) these queries have higher priority than other queries, and (3) a query may be killed if actual resource usage exceeds twice that estimated.

- The "OLTP" service class has queries that are short and have a fixed arrival rate. The queries are well-understood, and we have high confidence in how we expect them to behave. The objectives for these queries require the throughput to be above a certain transactions per second threshold and the average response time to be lower than another threshold, expressed in terms of milliseconds.
- The "Report" service class comprises medium-sized, roll-up report queries with an objective to complete all of the queries before a deadline. These queries are also well-understood.

3.1 Problem injection

Our workload management dashboard includes a policy control window where the admission, scheduling, and execution control policies are defined for each service class, as shown in Figure 3. The system is in steady state initially, with the OLTP queries running and meeting objectives, the report query batch running and on-track to meet its objective, and no ad hoc CEO queries in the system. Then an ad hoc CEO query arrives, is admitted, and starts executing. After a brief delay, system performance degrades and the OLTP queries are in danger of not meeting their objectives, as can be seen in the performance objective window in Figure 4, which shows how well each service class is meeting its objectives. Multiple CEO queries may execute simultaneously (causing even more contention), while there may be no such query in the system at other times. The workload management policies and thresholds may need adjusting with each increase or decrease in the number of these queries: resources that are reserved for these queries may be wasted when none is executing.

Figure 4: Performance objective window after problem injection shows the OLTP service class not meeting objectives.

3.2 Attempt at manual correction

Our demo challenges the participant to adjust the policies manually. In an attempt to diagnose the situation, the participant may view a system resource utilization window like that in Figure 5, which shows excessive resource utilization (memory and CPU) by the CEO queries. Note that diagnosing the cause of degraded performance is itself a challenge – it is not always obvious what or where the problem is. Our demonstration is also intended to help train database system administrators in this task.

There are multiple possible effective policy changes to reduce contention. For example, reducing the scheduling threshold (MPL) for the batch report queries will reduce resource contention. In addition, it may be necessary to kill or suspend some active report queries to achieve the newly lowered threshold. What makes this task particularly difficult is that re-allocating resources can have un-

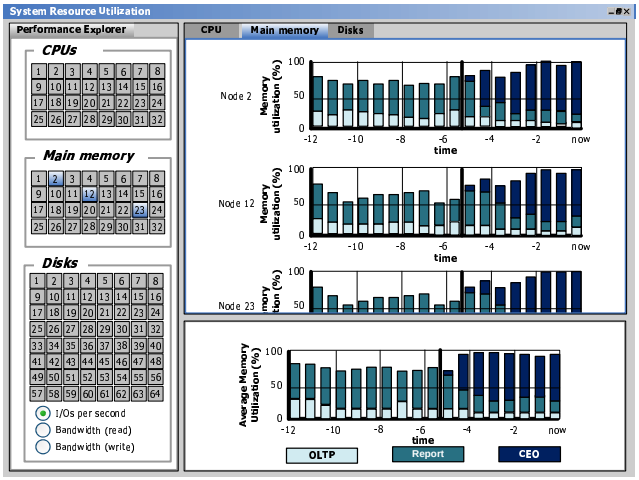

Figure 5: In the system resource utilization window, the administrator can see that after the arrival of a long-running, heavy-weight, CEO query, resource contention interferes with the rest of the workload.

Figure 6: Performance objective window after a run has completed. The demo participant can see how well the management actions they took worked with regard to service class objectives.

expected effects. For example, slowing down a long-running query may mean that it continues to occupy system resources for a longer time. The demonstration participant can use our interface to adjust various thresholds and policies and see the impact of their actions on system performance and service objectives.

In this simple example, one option is to suspend the report class temporarily to enable the CEO class to complete. Another option is to reduce the priority of the OLTP class. However, the best strategy is actually to reduce the scheduling MPL for the OLTP queries just a little (e. g., from 10 to 8 or 9) when the executive's ad hoc query starts executing. If the administrator lets the system load stay too high or else reduces the MPL too much, then the OLTP queries fail to meet their throughput objectives. If the administrator does not reduce the MPL enough, then the ad hoc query fails to meet its response time objective. If the human administrator takes too long to respond, then multiple objectives are missed.

In addition, as the CEO queries complete, the administrator should raise the MPL for the OLTP and/or Report queries, so that resources are not left idle. After a participant makes adjustments in the policy control window, the impact of those changes may be observed in the performance objective window. Figure 6 shows how the window might look at the end of a run. In the figure, one can see that although the OLTP and CEO queries did meet their throughput and response time objectives, the participant was not aggressive enough in throttling down the OLTP queries, and the Report workload missed its completion deadline. The participant can then compare when and which workload management actions they took, as well as the impact of those actions, to the actions that our automated system would have taken.

3.3 Policy control feedback loop

After the demo participant corrects the policies during execution of an entire workload, our demo replays the workload with our policy controller automatically adjusting the policies. The same GUI interfaces then show the participant's actions and their effects side-by-side with those of the policy controller. At the end of the workload, both are scored based on their ability to meet the workload objectives. The participant may then play a different scenario.

4. SUMMARY

Extreme diversity of resource requirements, the potential for show-stopping resource contention, and dynamically-arriving user queries that require significant changes to the workload management policies in effect all present significant challenges in managing a dynamic, mixed, operational BI workload. We believe that defining and implementing new policy control feedback loops can automate or greatly simplify many aspects of managing BI workloads. Our demonstration helps participants understand the impact of policies on mixed workloads and provides some positive examples of how to set policy choices. In addition, our demonstration framework allows us to create new workload scenarios so that we can study how our policy controller adapts to unexpected situations. This examination helps us to devise better policies and meta-policies.

5. ACKNOWLEDGMENTS

We would like to thank Stefan Kinauer for his help implementing the workload management user interface.

6. REFERENCES

[1] P. Gillin. BI @ the Speed of Business. *Computer World Technology*, December 2007.

[2] S. Krompass, H. Kuno, U. Dayal, and A. Kemper. Dynamic Workload Management for Very Large Data Warehouses: Juggling Feathers and Bowling Balls. In *Proc. of the 33rd Intl. Conf. on Very Large Data Bases (VLDB)*, 2007.

[3] S. Krompass, H. Kuno, J. Wiener, K. Wilkinson, U. Dayal, and A. Kemper. Managing Long-running Queries. In *To appear in: Proc. of the 12th Intl. Conf. on Extending Database Technology (EDBT)*, 2009.

[4] B. Niu, P. Martin, W. Powley, P. Bird, and R. Horman. Adapting Mixed Workloads to Meet SLOs in Autonomic DBMSs. In *Proc. of the 2007 Workshop on Self-Managing Database Systems (SMDB 2007)*, 2007.

[5] H. Pang, M. J. Carey, and M. Livny. Multiclass query scheduling in real-time database systems. *IEEE Trans. on Knowledge and Data Engineering*, 7(4), 1995.

[6] Teradata Dynamic Workload Manager User Guide, 2006.

[7] C. White. The Next Generation of Business Intelligence: Operational BI. *DM Review Magazine*, 2005.